Avoiding collisions
Cryptographic hash functions

Foundations of Cryptography
Computer Science Department
Wellesley College

Fall 2016

Table of contents

Hash functions

- *Hash functions* take arbitrary-length strings and *compress* them into shorter strings.
- The functions you studied in CS230 are examples where hashes are used to achieve $\mathcal{O}(1)$ lookup time in set implementations.
- Collisions are not good for data-retrieval complexity. They are disastrous in cryptographic applications.

Defining collisions

- A *collision* in a function H is a pair of distinct inputs x and x' such that $H(x) = H(x')$; we say that x and x' *collide* under H.
- A function H is *collision resistant* if it is infeasible for any probabilistic polynomial-time algorithm to find a collision in H.
- Typically, H is a compression function with infinite domain and finite range, so collisions must exist. Our goal is to make them hard to find.

James Garry, Fastlight Used with permission.

Keyed hash functions

- Formally, we deal with a *family* of hash functions indexed by a "key".
- More precisely, H will be a two-input function that takes as inputs a key s and a string x, and outputs a string $H^s(x) \stackrel{\text{def}}{=} H(s,x)$.
- It must be hard to find a collision in H^s for a randomly-generated s.

Formal definition of a hash function

Definition 5.1. A *hash function* is a pair of probabilistic polynomial-time algorithms (Gen, H) satisfying the following:

- Gen is a probabilistic algorithm which takes as input a security parameter 1^n and outputs a key s. We assume that 1^n is implicit in s.
- There exists a polynomial ℓ such that H takes as input a key s and a string $x \in \{0,1\}^*$ and outputs a string $H^s(x) \in \{0,1\}^{\ell(n)}$.

If H^s is defined only for inputs $x \in \{0,1\}^{\ell'(n)}$ and $\ell'(n) > \ell(n)$, then we say that (Gen, H) is a *fixed-length* hash function for inputs of length $\ell'(n)$.

Collision-finding experiments and collision resistance

The collision-finding experiment $\text{Hash-coll}_{\mathcal{A},\Pi}(n)$*:*

1. A key s is generated by running $\text{Gen}(1^n)$.
2. The adversary $\mathcal{A}$ is given s and outputs x, x'. (If Π is a fixed length hash function for inputs of length $\ell'(n)$ then we require $x, x' \in \{0,1\}^{\ell'(n)}$.)
3. The output of the experiment is defined to be 1 if and only if $x \neq x'$ and $H^s(x) = H^s(x')$. In such a case we say that $\mathcal{A}$ has found a collision.

Definition 5.2. A hash function $\Pi = (\text{Gen}, H)$ is *collision resistant* if for all probabilistic polynomial-time adversaries $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr[\text{Hash-coll}_{\mathcal{A},\Pi}(n) = 1] \leq \text{negl}(n).$$

A generic "birthday" attack

- There is a generic attack that finds collisions in *any* hash function. This attack implies a minimal output length needed for a hash function to be secure.
- Model the keyed hash function by a truly random function, $H : \{0,1\}^* \to \{0,1\}^\ell$.
- Choose q arbitrary inputs $x_1, \ldots, x_q \in \{0,1\}^{2\ell}$, compute $y_i := H(x_i)$ and check whether any two y_i are equal.

Analysis of the "birthday" problem

We did half of the analysis a couple of lectures ago:
Lemma A.15. Let $y_1, \ldots, y_q$ be q elements chosen uniformly at random from a set of of size N. The probability that there exists distinct i, j with $y_i = y_j$ is at most $\frac{q^2}{2N}$.

This time we prove:
Lemma A.16. Fix a positive integer N, and say $q \leq \sqrt{2N}$ elements $y_1, \ldots, y_q$ are chosen uniformly and independently at random from a set of size N. Then the probability that there exists distinct i, j with $y_i = y_j$ is at least $\frac{q(q-1)}{4N}$.

Remark. These lemmas imply that the birthday attack finds a collision with high probability using $q = \Theta(2^{\ell/2})$ hash-function evaluations. (Sorting the outputs and scanning for collisions requires an additional $\mathcal{O}(\ell \cdot 2^{\ell/2})$ time).

Proof of Lemma A.16

Lemma A.10. Fix a positive integer N, and say $q \leq \sqrt{2N}$ elements $y_1, \ldots, y_q$ are chosen uniformly and independently at random from a set of size N. Then the probability that there exists distinct i, j with $y_i = y_j$ is at least $\frac{q(q-1)}{4N}$.

Proof. Let Coll denote the event of a collision and let NoColl_i be the event that there is no collision among $y_1, \ldots, y_i$. Then $\text{NoColl}_q = \overline{\text{Coll}}$ is the event that there is no collision at all.

Calculating $\Pr[\mathsf{NoColl}_q]$

If NoColl_q occurs then NoColl_i must also have occurred for all $i \leq q$. Thus,

$\Pr[\mathsf{NoColl}_q] = \Pr[\mathsf{NoColl}_1] \cdot \Pr[\mathsf{NoColl}_2 \mid \mathsf{NoColl}_1] \cdots \Pr[\mathsf{NoColl}_q \mid \mathsf{NoColl}_{q-1}]$.

Certainly $\Pr[\mathsf{NoColl}_1] = 1$, and if $\{y_1, \ldots, y_i\}$ are distinct, the probability that y_{i+1} does not collide with any of these values is $1 - \frac{i}{N}$. In other words,

$$\Pr[\mathsf{NoColl}_{i+1} \mid \mathsf{NoColl}_i] = 1 - \frac{i}{N},$$

so

$$\Pr[\mathsf{NoColl}_q] = \prod_{i=1}^{q-1}\left(1 - \frac{i}{N}\right).$$

Math 115 returns with a vengeance

From the first two terms of the Taylor series expansion of e^x, $1 - \frac{i}{N} \leq e^{-i/N}$, so

$$\begin{aligned}\Pr[\mathsf{NoColl}_q] &= \prod_{i=1}^{q-1}\left(1 - \frac{i}{N}\right) \\ &\leq \prod_{i=1}^{q-1} e^{-i/N} \\ &= e^{-\sum_{i=1}^{q-1}(i/N)} = e^{-q(q-1)/2N}.\end{aligned}$$

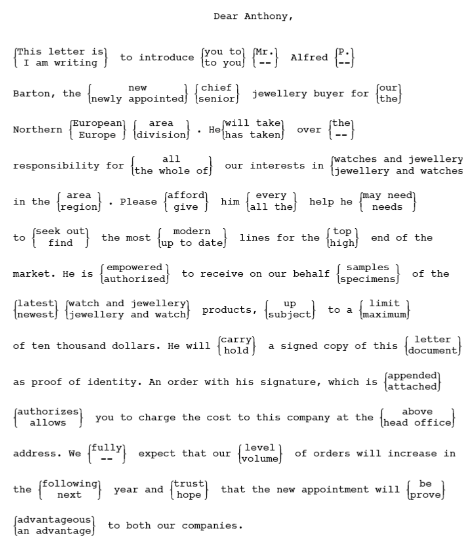
We conclude that

$$\Pr[\mathsf{Coll}] = 1 - \Pr[\mathsf{NoColl}_q] \geq 1 - e^{-q(q-1)/2N} \geq \frac{q(q-1)}{4N}.$$

*The last line uses the fact that for all $0 \leq x \leq 1$, $e^x \leq 1 - x/2$.

Birthday attacks matter

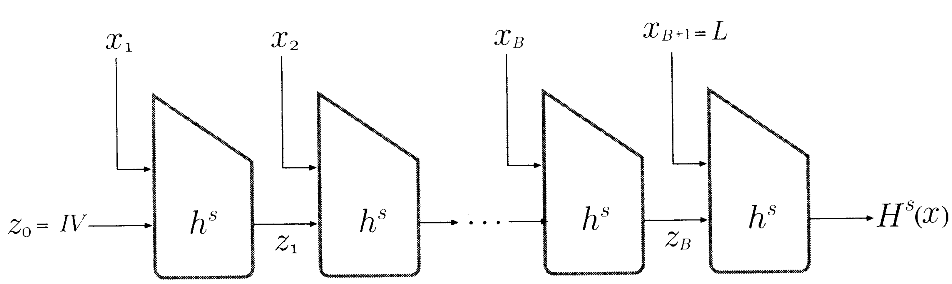
- A hash function designed with output length 128 bits seems secure since running 2^{128} steps to find a collision seems infeasible.
- However, the generic birthday attack requires only 2^{64} steps, large but not impossible. Furthermore, evil doers may use collisions to their advantage.


Dear Anthony,

{This letter is / I am writing} to introduce {you to / to you} {Mr. / --} Alfred {P. / --} Barton, the {new / newly appointed} {chief / senior} jewellery buyer for {our / the} Northern {European / Europe} {area / division}. He {will take / has taken} over {the / --} responsibility for {all / the whole of} our interests in {watches and jewellery / jewellery and watches} in the {area / region}. Please {afford / give} him {every / all the} help he {may need / needs} to {seek out / find} the most {modern / up to date} lines for the {top / high} end of the market. He is {empowered / authorized} to receive on our behalf {samples / specimens} of the {latest / newest} {watch and jewellery / jewellery and watch} products, {up / subject} to a {limit / maximum} of ten thousand dollars. He will {carry / hold} a signed copy of this {letter / document} as proof of identity. An order with his signature, which is {appended / attached} {authorizes / allows} you to charge the cost to this company at the {above / head office} address. We {fully / --} expect that our {level / volume} of orders will increase in the {following / next} year and {trust / hope} that the new appointment will {be / prove} {advantageous / an advantage} to both our companies.

Constructing collision-resistant hashes: Merkle-Damgärd transform

The Merkle-Damgärd transform is widely used to convert fixed-length hash functions to full-fledged hashes.



Assume we are given a fixed-length collision-resistant hash function, (Gen, h), that compresses its input length by half; i.e., $\ell'(n) = 2\ell(n)$. We construct a collision-resistant hash (Gen, H) that maps any length inputs to outputs of length $\ell(n)$.

The construction

Construction 5.3.
Let (Gen, h) be a fixed-length collision-resistent hash function for inputs of length $2\ell(n)$ and with output length $\ell(n)$. Construct a variable-length hash function (Gen, H) as follows:

- Gen: Remains unchanged.
- H: On input a key s and a string $x \in \{0,1\}^*$ of length $L < 2^{\ell(n)}$, do the following:
  1. Set $B := \lceil \frac{L}{\ell} \rceil$. Pad x with zeros so its length is a multiple of ℓ. Parse the padded result as the sequence of ℓ-bit blocks $x_1, \ldots, x_B$. Set $x_{B+1} := L$, where L is encoded using exactly ℓ bits.
  2. Set $z_0 := 0^\ell$.
  3. For $i = 1, \ldots, B+1$, compute $z_i := h^s(z_{i-1} \| x_i)$.
  4. Output z_{B+1}.

Collision-resistance results

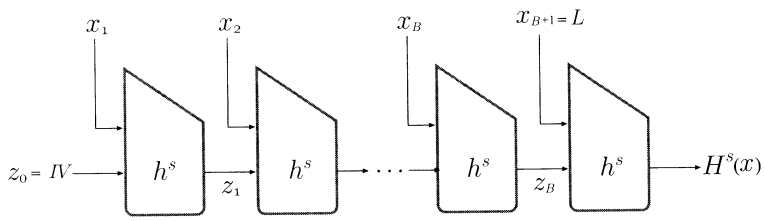
Theorem 5.4. If (Gen, h) is a fixed-length collision-resistant hash function, then (Gen, H) is a collision-resistent hash function.

Proof. We show that for any s, a collision in H^s yields a collision in h^s.

Let x and x' be two different strings of respective lengths L and L' such that $H^s(x) = H^s(x')$. Let $x_1, \ldots, x_B$ be the B blocks of the padded x, and let $x'_1, \ldots, x'_{B'}$ be the B' blocks of the padded x'. Recall that $x_{B+1} = L$ and $x'_{B'+1} = L'$.

The first of two cases

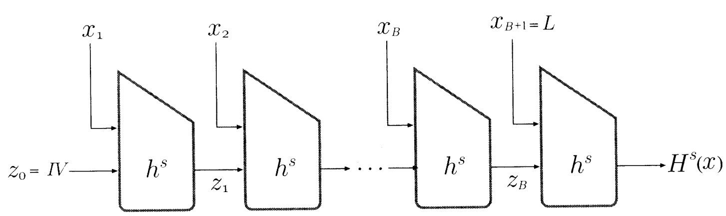
Case 1: $L \neq L'$. The last step of the computation of $H^s(x)$ is $z_{B+1} := h^s(Z_B \| L)$ and the last step of the computation of $H^s(x')$ is $z'_{B'+1} := h^s(z'_{B'} \| L')$. Since $H^s(x) = H^s(x')$ it follows that $h^s(z_B \| L) = h^s(z'_{B'} \| L')$.



Collision city: $z_B \| L \neq z'_{B'} \| L'$, while $L \neq L'$.

The second of two cases

Case 2: $L = L'$. Then $B = B'$ and $x_{B+1} = x'_{B+1}$. Let z_i, z'_i be intermediate values of x, x' during the computation of $H^s(x), H^s(x')$ respectively.



There must be at least one index i such that $x_i \neq x'_i$. Let i^* be the highest index for which $z_{i^*-1} \| x_{i^*} \neq z'_{i^*-1} \| x'_{i^*}$. If $i^* = B+1$ then $z_B \| x_{B+1}$ and $z'_B \| x'_{B+1}$ are different strings that collide for h^s, because

$$h^s(z_B \| x_{B+1}) = z_{B+1} = H^s(x) = H^s(x') = z'_{B+1} = h^s(z'_B \| x'_{B+1}).$$

If $i^* \leq B$, then the maximality of i^* implies $z_{i*} = z'_{i*}$. Once again, $z_{i^*-1} \| x_{i^*} \neq z'_{i^*-1} \| x'_{i^*}$ are two strings that collide for h^s.

Collision-resistant hash functions in practice

- SHA-1, a common iterated hash, inputs a string of any length up to $2^{64} - 1$, and produces an output of length 160 bits.*
- SHA-1 starts with a *compression function* that compresses fixed-length inputs by a small amount. A Merkle-Damgärd transform is applied to obtain a collision-resistant hash function.
- Think of $2^{2^{64}}$ pigeons (the 2^{64}-bit strings) roosting in 2^{160} pigeon holes. Things have got to be crowded. However, nobody has found a hole with two or more pigeons.

*MD4, MD5,and SHA are ancestors (not be trusted); SHA-256, SHA-384, and SHA-512 descendants.

Big MACs

- We built our MACs from pseudorandom functions. Here we will see another approach that relies on collision-resistant hashing along with message authentication code.
- A another common* approach is based on collision-resistant hash functions constructed using the Merkel-Damgärd transformation. We will save that for next time.

*Not to mention more efficient.

Hash-and-MAC

The idea is simple: A long message m is hashed down to a fixed-length string $H^s(m)$ using a collision-resistant has function, then a fixed-length MAC is applied.

Construction 5.5 Let $\Pi = (\mathsf{Mac}, \mathsf{Vrfy})$ be a MAC for message of length $\ell(n)$, and let $\Pi_H = (\mathsf{Gen}_H, H)$ be a hash function with output of length $\ell(n)$. Construct a Mac $\Pi' = (\mathsf{Gen}', \mathsf{Mac}', \mathsf{Vrfy}')$ for arbitrary-length messages as follows

- Gen': On input 1^n, choose uniform $k \in \{0,1\}$ and run $\mathsf{Gen}_{\mathsf{H}}(1^n)$ to obtain a key s. The key is $k' := \langle k, s\rangle$.
- Mac': On input $\langle k, s\rangle$ and message $m \in \{0,1\}^*$, output the tag $t \leftarrow \mathsf{Mac}_k(H^s(m))$.
- Vrfy': On input a key $\langle k, s\rangle$, a message $m \in \{0,1\}^*$, and a MAC tag t, output 1 if and only if $\mathsf{Vrfy}_k(H^s(m), t) \stackrel{?}{=} 1$.

If Π is secure, then so is Π'

Theorem 5.6. If Π is a secure MAC for message of length ℓ and Π_H is collision resistant, then Contruction 5.5 is a secure MAC for arbitrary length messages.

Proof. let Π' denote Construction 5.5, and let $\mathcal{A}'$ be a PPT adversary attacking Π'. In $\mathsf{Mac\text{-}forge}_{\mathcal{A}',\Pi'}$, let $k' = \langle k, s\rangle$ denote the MAC key, $\mathcal{Q}$ denote the set of messages whose tags were requested by $\mathcal{A}'$, and (m^*, t) be the final output of $\mathcal{A}'$.

Assume WLOG that $m^* \notin \mathcal{Q}$ and define coll to be the event that there is an $m \in \mathcal{Q}$ for which $H^s(m^*) = H^s(m)$. We have

$$\begin{aligned}
&\Pr[\mathsf{Mac\text{-}forge}_{\mathcal{A}',\Pi'}(n) = 1] \\
&= \Pr[\mathsf{Mac\text{-}forge}_{\mathcal{A}',\Pi'}(n) = 1 \wedge \mathsf{coll}] + \Pr[\mathsf{Mac\text{-}forge}_{\mathcal{A}',\Pi'}(n) = 1 \wedge \overline{\mathsf{coll}}] \\
&\le \Pr[\mathsf{coll}] + \Pr[\mathsf{Mac\text{-}forge}_{\mathcal{A}',\Pi'}(n) = 1 \wedge \overline{\mathsf{coll}}].
\end{aligned}$$

We show both terms above are negligible.

Algorithm for finding a collision in Π_H

Algorithm $\mathcal{C}$ The algorithm is given s as an input.

- Choose uniform $k \in \{0,1\}^n$.
- Run $\mathcal{A}'(1^n)$. When $\mathcal{A}'$ requests a tag on $m_i \in \{0,1\}^*$, return $t_i \leftarrow \text{Mac}_k(H^s(m_i))$.
- When $\mathcal{A}'$ outputs (m^*, t), then if there exists an i for which $H^s(m^*) = H^s(m_i)$, output (m^*, m_i).

When the input s to $\mathcal{C}$ is generated by running $\text{Gen}_H(1^n)$, then the view of $\mathcal{A}'$ is distributed identically to its view in $\text{Mac-forge}_{\mathcal{A}',\Pi,}(n)$. Since $\mathcal{C}$ outputs a collision exactly when coll occurs, we have

$$\Pr[\text{Hash-coll}_{\mathcal{C},\Pi_H}(n) = 1] = \Pr[\text{coll}].$$

Since Π_H is collision resistant, we conclude that Pr[coll] is negligible.

We show $\Pr[\textit{Mac-forge}_{\mathcal{A}',\Pi'}(n) = 1 \wedge \overline{\textit{coll}}]$ *is negligible*

Algorithm $\mathcal{A}$

- Compute $\text{Gen}_H(1^n)$ to obtain s.
- Run $\mathcal{A}'(1^n)$. When $\mathcal{A}'$ requests a tag on $m_i \in \{0,1\}^*$, then (1) compute $\hat{m}_i := H^s(m_i)$; (2) obtain a tag t_i on $\hat{m}_i$ from the MAC oracle; and (3) give t_i to $\mathcal{A}'$.
- When $\mathcal{A}'$ outputs (m^*, t), then output $(H^s(m^*), t)$.

In the experiment $\text{Mac-forge}_{\mathcal{A},\Pi}(n)$, the view of $\mathcal{A}'$ when run as a subroutine by $\mathcal{A}$ is distributed identically to its view in experiment $\text{Mac-forge}_{\mathcal{A}',\Pi,}(n)$. Whenever both $\text{Mac-forge}_{\mathcal{A}',\Pi,}(n) = 1$ and coll do not occur, $\mathcal{A}$ outputs a valid forgery. Therefore,

$$\Pr[\text{Mac-forge}_{\mathcal{A},\Pi}(n) = 1] = \Pr[\text{Mac-forge}_{\mathcal{A}',\Pi'}(n) = 1 \wedge \overline{\text{coll}}],]$$

and security of Π implies that the former probability is negligible. □